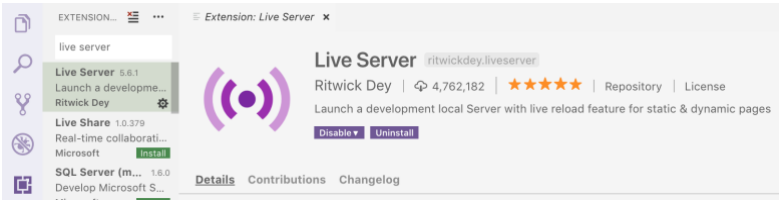
# CREATE A WEB AR TREASURE HUNT GAME

## Software/ Programs Needed

- Microsoft Visual Code
- Extension in VS Code (Live Server – Ritwick Dey)




## Workshop Overview

The objective of this workshop is to teach A-frame and AR.js basics and how to build a simple AR Treasure Hunt Game.


## Workshop Steps

1. **What is WebVR?**

WebVR is an experimental JavaScript application programming interface (API) that provides support for virtual reality devices, such as the HTC Vive, Oculus Rift, Google Cardboard or OSVR in a web browser.

This API is designed with the following goals in mind:

- Detect available Virtual Reality devices.
- Query the devices capabilities.
- Poll the device's position and orientation.
- Display imagery on the device at the appropriate frame rate.

WebVR makes it possible to experience VR in your browser. The goal is to make it easier for everyone to get into VR experiences, no matter what device you have.
(https://webvr.info/)

## 2. What is A-Frame?

A-Frame is an open-source web framework for building virtual reality (VR) experiences. It is maintained by developers from Supermedium (Diego Marcos, Kevin Ngo) and Google (Don McCurdy). A-Frame is an entity component system framework for Three.js where developers can create 3D and WebVR scenes using HTML. HTML provides a familiar authoring tool for web developers and designers while incorporating a popular game development pattern used by engines such as Unity.
(https://aframe.io/)

## 3. What is AR.js?

AR.js is a JavaScript framework providing an efficient Augmented Reality solution on the Web acting as a port of AR Toolkit, while leveraging other packages like a-frame and three.js.

## 4. Markers

An important feature of AR.js is the possibility to use custom markers, the default type is 'pattern'.

AR.js uses artoolkit, and so it is marker based. artoolkit is a software with years of experience doing augmented reality.
It supports a wide range of markers: multiple types of markers pattern/barcode multiple independent markers at the same time, or multiple markers acting as a single marker, up to you to choose.

Criteria for creating your own marker -

- They must be square in shape.
- They cannot have white/transparent areas, only black and light grey (e.g. #F0F0F0)
- They have to contain simple text, like one letter, a number, or a symbol.

To create your own markers, go to (https://jeromeetienne.github.io/AR.js/three.js/examples/marker-training/examples/generator.html)

Two default markers are – hiro & kanji


## 5. Installation Guide

**Windows**

1. Download Node.js (https://nodejs.org/en/) (LTS Version Recommended)

2. Open Node.js Command Prompt and type to display versions.
 node -v
 npm –v

 3. Install obj to gltf converter
 npm install -g obj2gltf

**Mac**

1. Install Xcode from App Store or update to the latest version if already installed (https://itunes.apple.com/ae/app/xcode/id497799835?mt=12)

2. Open Terminal and Install Homebrew (https://brew.sh) using the following command
  /usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

3. Install node
  brew install node

 Check node & npm version
  node -v
  npm -v

4. Install obj to gltf converter

npm install -g obj2gltf

Convert a .obj to .gltf
obj2gltf -i Pyra.obj -o Pyra.gltf

## 6. Code

Include the JS Build
To include A-Frame & AR.js in an HTML file, we drop a <script> tag pointing to the CDN build:

```
<head>
 <script src="https://aframe.io/releases/0.9.2/aframe.min.js"></script>
 <script
src="https://cdn.rawgit.com/jeromeetienne/AR.js/1.7.1/aframe/build/aframe-ar.js"></script>
</head>
```

To include AR.js, you need to include aframe-ar.js. Then you initialize ar.js in.

```
<a-scene embedded arjs>
```

Then you tell A-Frame that you want arjs to control the camera. For that, you just add

```
<a-marker-camera preset='hiro'></a-marker-camera>
```

**Step 1 – Default Object**

```
<a-box>/<a-triangle>/<a-ring>/<a-sphere>/<a-tetrahedron>/<a-cylinder>/<a-sphere>

<a-box position='0 0.5 0' scale='0.5 0.5 0.5' rotation='0 0 0' material='color: yellow;'></a-box>
```

**Entire Code (1. Default_Object.html) –**

```
<!doctype HTML>
<html>
<script src="https://aframe.io/releases/0.9.2/aframe.min.js"></script>
<script
src="https://cdn.rawgit.com/jeromeetienne/AR.js/1.7.1/aframe/build/aframe-ar.js"></script>
  <body>
    <a-scene embedded arjs>
      <a-marker preset="hiro">
```

```
        <a-triangle position='0.5 0 0' scale='0.8 0.8 0.8' rotation='-90 45 0'
material='color: white; opacity: 0.9;' ></a-triangle>
      </a-marker>
      <a-entity camera></a-entity>
    </a-scene>
  </body>
</html>
```


**Step 2- Using an image as an object**

```
        <a-sphere>

        <a-box position='0 0.5 0' scale='0.5 0.5 0.5' rotation='0 0 0' material='color:
yellow;' material='opacity: 0.5;'></a-box>
```

**Animation**

The animation component lets us animate and tween values including:

- Component values (e.g., position, visible)
- Component property values (e.g., light.intensity)

We can also tween values directly for better performance versus going through.setAttribute, such as by animating values:

- On the object3D (e.g., object3D.position.y, object3D.rotation.z)
- Directly within a component (e.g., components.material.material.color, components.text.material.uniforms.opacity.value)

*<a-animation attribute="scale" dur="2000" from= "0 0 0" to="5 5 5" direction='alternate-reverse' easing= "ease-in-out-circ" repeat="indefinite"></a-animation>*

(https://aframe.io/docs/0.9.0/components/animation.html)

**Entire Code (2. Image_as_an_Object.html) –**

```
<!-- Augmented Reality on the Web in 10 lines of html!
https://github.com/jeromeetienne/ar.js -->
<script src="https://aframe.io/releases/0.9.2/aframe.min.js"></script>
<script
src="https://rawgit.com/jeromeetienne/ar.js/master/aframe/build/aframe-ar.js"></script>
<script>THREEx.ArToolkitContext.baseURL =
'https://rawgit.com/jeromeetienne/ar.js/master/three.js/'</script>
<body style='margin : 0px; overflow: hidden;'>
    <a-scene embedded artoolkit='sourceType: webcam;'>
    <a-sphere src="https://raw.githubusercontent.com/aframevr/sample-assets/master/assets/images/space/earth_atmos_4096.jpg" radius="0.5"
position="0 0.5 0" segments-height="53">
        <a-animation attribute="scale"
            dur="2000"
            from= "0 0 0"
            to="5 5 5"
            direction='alternate-reverse'
            easing= "ease-in-out-circ"
            repeat="indefinite"></a-animation>
    </a-sphere>
            <a-marker-camera preset='hiro'></a-marker-camera>
    </a-scene>
</body>
```

**Step 3 – Treasure Hunt**

Adding Markers & 3D models to the A-Frame HTML Template
1. Add the src () of the 3D asset to the node of the template.
Using .obj file

*<a-asset-item id="pyra-obj" src="builder-models/Pyra.obj"></a-asset-item>*

Using .gltf file

*<a-asset-item id="pyra" src="builder-markers/Pyra.gltf"></a-asset-item>*

Add the src () of the Marker data file to the node of the template.

*<a-marker id ="pyra-marker" type="pattern" url="builder-markers/pyra.patt">*

In order to “link” the Marker and the 3D model, add an node with a reference to the 3D model inside the node created in the previous step.

**Using .obj file**

*<a-entity id="pyra" obj-model="obj: #pyra-obj;" material="color: green" rotation="0 180 0" position="0 0 0.5" scale="0.15 0.15 0.15"></a-entity>*

**Using .gltf file**

```
<a-entity  rotation="90 -45 45" position="0 0 0" scale="1 1 1" gltf-model="#pyra"></a-entity>
```

**Implementing Speech Bubble Dialogue Interaction**

Tap functionality by defining a new component called accepts-clicks:

```
AFRAME.registerComponent('accepts-clicks', {
  init: function() {
    this.el.addEventListener('touchend', handleClickEvent);
    this.el.addEventListener('click', handleClickEvent);
  }
});
```

and then adding it as an attribute to:

```
<a-scene embedded arjs accepts-clicks>
```

Based on our design of the treasure hunt, only one Marker would be visible on the screen at once, so all we needed to do was loop through the Builders every time the screen was tapped and use the dialogue of the Builder whose object3D.visible is true.

```
function handleClickEvent() {
  for (var i = 0; i < builders.length; i++) {
    var builder = builders[i];
    var builderMarker = document.querySelector("#" + builder.name + "-marker");
    if (builderMarker && builderMarker.object3D.visible) {
      if (searchForBuilderTool(builder)) {
        toggleSpeechBubble(builder.successDialogue);
      } else {
        toggleSpeechBubble(builder.dialogue);
      }
      break;
    }
  }
}
```

Defined a “tick” handler (which gets called for every render loop of A-Frame).

```
AFRAME.registerComponent('accepts-clicks', {
  init: function() {
    this.el.addEventListener('touchend', handleClickEvent);
    this.el.addEventListener('click', handleClickEvent);
  },
  tick: function() {
    hideSpeechBubbleIfNoMarker();
  }
});
```

*The function hideSpeechBubbleIfNoMarker() uses a similar test with object3D.visible to determine if the speech bubble should be hidden. If all Builders are hidden, also hide the speech bubble, but if any Builder is visible, keep the speech bubble.*

```
function hideSpeechBubbleIfNoMarker() {
  var shouldHide = true;
  for (var i = 0; i < builders.length; i++) {
    var builderMarker = document.querySelector("#" + builders[i].name + "-
marker");
    if (builderMarker && builderMarker.object3D.visible) {
      shouldHide = false;
      break;
    }
  }
  // hide speech bubble
};
```

**Keeping Track of Player Inventory**

The final piece to our treasure hunt was keeping track of which treasures the player had found. We defined several classes: Builder, Tool (aka treasures), and UserState.

The Builder class had fields for name, tool, initial dialogue, and alternate dialogue for after the user has found the treasure. The “tool” field was the treasure associated with that Builder. The Tool class had fields for name and dialogue. And finally, UserState had an array containing the inventory of treasures that the player had clicked on. Every time the player clicked a tool, that tool would be added to the UserState. And every time the player clicked a Builder, we would check whether the tool associated with that Builder was in the UserState. If so, we would show the alternate dialogue.

**Entire Code**

**userState.js**

```
function UserState() {
    this.tools = [];
}

UserState.prototype.addTool = function(tool) {
    this.tools.push(tool);
}

UserState.prototype.hasBuilderTool = function(builder) {
    return builder.tool && this.tools.includes(builder.tool.name);
}
```

```
var userState = new UserState();
```

**models.js**

```
var builders = [],
    tools = [];

function ARModel(name, dialogue) {
    //we can make name link to the el id to find it on click?
    this.name = name;
    this.dialogue = dialogue;


}

ARModel.prototype.speak = function() {
    return this.dialogue;
}

//Builder model
function Builder(name, dialogue, tool, successDialogue) {
    ARModel.call(this, name, dialogue);
    this.tool = tool;
    this.successDialogue = successDialogue;
}

Builder.prototype = Object.create(ARModel.prototype);

//Tool model
function Tool(name, dialogue) {
    ARModel.call(this, name, dialogue);
}

Tool.prototype = Object.create(ARModel.prototype);

function initiateModels() {
    var buildersArray = [
      {
        name: 'pyra',
        dialogue: 'Hi there, I\'m Pyra! I\'ve lost my hammer. Let me know if you
see it!',
        tool: new Tool('hammer', 'You have found Pyra\'s hammer!'),
        successDialogue: 'Thanks for my hammer!'
      },
      {
        name: 'biggie',
        dialogue: 'Hey, I\'m Biggie! I left my blocks somewhere in the office... can
you help me find it?',
        tool: new Tool('blocks', 'You have found Biggie\'s blocks!'),
```

```
            successDialogue: 'My blocks have been found!'
          }
        ];

        buildersArray.forEach(function(builder){
            builders.push(new Builder(builder.name, builder.dialogue, builder.tool,
    builder.successDialogue));
            if (builder.tool) tools.push(builder.tool);
        });

        console.log('builders', builders);
        console.log('tools', tools)
    }

    initiateModels();
```

**interaction.js**

```
AFRAME.registerComponent('accepts-clicks', {
  init: function() {
    this.el.addEventListener('touchend', handleClickEvent);
    this.el.addEventListener('click', handleClickEvent);
  },
  tick: function() {
    hideSpeechBubbleIfNoMarker();
  }
});

function hideSpeechBubbleIfNoMarker() {
  var speechBubble = document.querySelector(".speech-bubble");
  if (speechBubble === null) return;
  if (speechBubble.style.display === 'none' || !speechBubble.style.display) return;

  var shouldHide = true;
  builders.forEach(function(builder){
    var builderMarker = document.querySelector("#" + builder.name + "-marker");
    if (builderMarker && builderMarker.object3D.visible) shouldHide = false;
  });

  tools.forEach(function(tool){
    var toolMarker = document.querySelector("#" + tool.name + "-marker");
    if (toolMarker && toolMarker.object3D.visible) shouldHide = false;
  });

  if (shouldHide) speechBubble.style.display = 'block';
};

function handleClickEvent() {
```

```
  builders.forEach(function(builder) {
    var builderMarker = document.querySelector("#" + builder.name + "-marker");
    if (builderMarker && builderMarker.object3D.visible) {
      if (searchForBuilderTool(builder)){
        toggleSpeechBubble(builder.successDialogue);
      } else {
        toggleSpeechBubble(builder.dialogue);
      }
    }
  });

  tools.forEach(function(tool){
    var toolMarker = document.querySelector("#" + tool.name + "-marker");
    if (toolMarker && toolMarker.object3D.visible) {
      toggleSpeechBubble(tool.dialogue);
      if (!userState.hasBuilderTool(tool)) userState.addTool(tool);
    }
  });
}

function toggleSpeechBubble(dialogue) {
  var speechBubble = document.querySelector(".speech-bubble");
        if (speechBubble.style.display === 'none' || !speechBubble.style.display) {
                speechBubble.innerHTML = dialogue;
                speechBubble.style.display = 'block';
        } else {
                speechBubble.style.display = 'none';
        }
};

function searchForBuilderTool(builder) {
  return userState.tools.some(function(tool) {
    return tool.name === builder.tool.name;
  });
};
```

**style.css**

```
.speech-bubble {
        position: absolute;
        width: 80%;
        margin: 0 auto;
        bottom: 50px;
        right: 10px;
        left: 10px;
    min-height: 20%;
        background: #004431;
        color: #0f4;
        font-size: 28px;
```

```
    border-radius: .4em;
        padding: 20px;
    display: none;
}

.speech-bubble:after {
        content: '';
        position: absolute;
        bottom: 0;
        left: 50%;
        width: 0;
        height: 0;
        border: 50px solid transparent;
        border-top-color: #004431;
        border-bottom: 0;
        border-left: 0;
        margin-left: -50px;
        margin-bottom: -50px;
}

#arjsDebugUIContainer {
    display: none;
}

.a-enter-vr {
        display: none;
}
```

**Treasure_Hunt.html**

```
<html>
  <head>
    <link rel="stylesheet" href="style.css">
  </head>
<script src="https://aframe.io/releases/0.9.2/aframe.min.js"></script>
<script src="https://jeromeetienne.github.io/AR.js/aframe/build/aframe-
ar.js"></script>
<script src="js/models.js"></script>
<script src="js/userState.js"></script>
<script src="js/interaction.js"></script>
<body style='margin : 0px; overflow: hidden;'>
    <div class='speech-bubble' style="z-index:1; font-size:50px"></div>

  <a-scene embedded arjs cursor="rayOrigin: mouse" accepts-clicks>

<!-- below is for content -->
<a-assets>
<!-- id="file name“ src="file url" -->
  <a-asset-item id="pyra" src="builder-markers/Pyra.gltf"></a-asset-item>
```

```
  <a-asset-item id="blocks" src="builder-markers/BuildingBlocks.gltf"></a-asset-item>
  <a-asset-item id="hammer" src="builder-markers/Hammer.gltf"></a-asset-item>
  <a-asset-item id="biggie" src="builder-markers/biggie.gltf"></a-asset-item>
</a-assets>

<!-- scale values allow you to adjust the file's display size, gltf-model="#your file name" -->

    <a-marker id ="pyra-marker" type="pattern" url="builder-markers/pyra.patt">
    <a-entity  rotation="0 180 0" position="0 0 0.5" scale="0.15 0.15 0.15" gltf-model="#pyra"></a-entity>
    </a-marker>

    <a-marker id ="blocks-marker" type="pattern" url="builder-markers/blocks.patt">
    <a-entity  rotation="90 -45 45" position="0 0 0" scale="1 1 1" gltf-model="#blocks"></a-entity>
    </a-marker>

    <a-marker preset='hiro' id ="hammer-marker">
    <a-entity  rotation="90 -45 0" position="0 0 0" scale="1 1 1"  gltf-model="#hammer"></a-entity>
    </a-marker>

    <a-marker preset='kanji' id="biggie-marker">
    <a-entity  rotation="0 180 0" position="0 0 0.5" scale="0.15 0.15 0.15" gltf-model="#biggie"></a-entity>
    </a-marker>

    <a-entity camera></a-entity>
    </a-scene>
</body>
```